How Transferable are Neural Networks in NLP Applications?

Lili Mou,[1] **Zhao Meng,**[1] **Rui Yan,**[2] **Ge Li,**[1,†] **Yan Xu,**[1,∗] **Lu Zhang,**[1] **Zhi Jin**[1,†]

[1]Key Laboratory of High Confidence Software Technologies (Peking University), MoE, China
Institute of Software, Peking University, China   [†]Corresponding authors
[2]Insitute of Computer Science and Technology of Peking University, China

{doublepower.mou,rui.yan.peking}@gmail.com,zhaomeng.pku@outlook.com
{lige,xuyan14,zhanglu,zhijin}@sei.pku.edu.cn

Abstract

Transfer learning is aimed to make use of valuable knowledge in a *source* domain to help model performance in a *target* domain. It is particularly important to neural networks, which are very likely to be overfitting. In some fields like image processing, many studies have shown the effectiveness of neural network-based transfer learning. For neural NLP, however, existing studies have only casually applied transfer learning, and conclusions are inconsistent. In this paper, we conduct systematic case studies and provide an illuminating picture on the transferability of neural networks in NLP.[1]

1 Introduction

Transfer learning, or sometimes known as domain adaptation,[2] plays an important role in various natural language processing (NLP) applications, especially when we do not have large enough datasets for the task of interest (called the *target* task $\mathcal{T}$). In such scenarios, we would like to transfer or adapt knowledge from other domains (called the *source* domains/tasks $\mathcal{S}$) so as to mitigate the problem of overfitting and to improve model performance in $\mathcal{T}$. For traditional feature-rich or kernel-based models, researchers have developed a variety of elegant methods for domain adaptation; examples include EasyAdapt (Daumé III, 2007; Daumé III et

al., 2010), instance weighting (Jiang and Zhai, 2007; Foster et al., 2010), and structural correspondence learning (Blitzer et al., 2006; Prettenhofer and Stein, 2010).

Recently, deep neural networks are emerging as the prevailing technical solution to almost every field in NLP. Although capable of learning highly nonlinear features, deep neural networks are very prone to overfitting, compared with traditional methods. Transfer learning therefore becomes even more important. Fortunately, neural networks can be trained in a transferable way by their incremental learning nature: we can directly use trained (tuned) parameters from a source task to initialize the network in the target task; alternatively, we may also train two tasks simultaneously with some parameters shared. But their performance should be verified by empirical experiments.

Existing studies have already shown some evidence of the transferability of neural features. For example, in image processing, low-level neural layers closely resemble Gabor filters or color blobs (Zeiler and Fergus, 2014; Krizhevsky et al., 2012); they can be transferred well to different tasks. Donahue et al. (2014) suggest that high-level layers are also transferable in general visual recognition; Yosinski et al. (2014) further investigate the transferability of neural layers in different levels of abstraction.

Although transfer learning is promising in image processing, conclusions appear to be less clear in NLP applications. Image pixels are low-level signals, which are generally continuous and less related to semantics. By contrast, natural language tokens

---

[∗]Yan Xu is currently a research scientist at Inveno Co., Ltd.
[1]Code released on https://sites.google.com/site/transfernlp/
[2]In this paper, we do not distinguish the conceptual difference between *transfer learning* and *domain adaptation*. *Domain*—in the sense we use throughout this paper—is defined by datasets.

are discrete: each word well reflects the thought of humans, but neighboring words do not share as much information as pixels in images do. Previous neural NLP studies have casually applied transfer techniques, but their results are not consistent. Collobert and Weston (2008) apply multi-task learning to SRL, NER, POS, and CHK,[3] but obtain only 0.04–0.21% error reduction[4] (out of a base error rate of 16–18%). Bowman et al. (2015), on the contrary, improve a natural language inference task from an accuracy of 71.3% to 80.8% by initializing parameters with an additional dataset of 550,000 samples. Therefore, more systematic studies are needed to shed light on transferring neural networks in the field of NLP.

Our Contributions

In this paper, we investigate the question *"How transferable are neural networks in NLP applications?"*

We distinguish two scenarios of transfer: (1) transferring knowledge to a semantically similar/equivalent task but with a different dataset; (2) transferring knowledge to a task that is semantically different but shares the same neural topology/architecture so that neural parameters can indeed be transferred. We further distinguish two transfer methods: (1) using the parameters trained on $\mathcal{S}$ to initialize $\mathcal{T}$ (INIT), and (2) multi-task learning (MULT), i.e., training $\mathcal{S}$ and $\mathcal{T}$ simultaneously. (Please see Sections 2 and 4). Our study mainly focuses on the following research questions:

RQ1: How transferable are neural networks between two tasks with similar or different semantics in NLP applications?

RQ2: How transferable are different layers of NLP neural models?

RQ3: How transferable are INIT and MULT, respectively? What is the effect of combining these two methods?

We conducted extensive experiments over six datasets on classifying sentences and sentence pairs. We leveraged the widely-used convolutional neural network (CNN) and long short term memory (LSTM)-based recurrent neural network (RNN) as our models.

Based on our experimental results, we have the following main observations, some of which are unexpected.

- Whether a neural network is transferable in NLP depends largely on how semantically similar the tasks are, which is different from the consensus in image processing.
- The output layer is mainly specific to the dataset and not transferable. Word embeddings are likely to be transferable to semantically different tasks.
- MULT and INIT appear to be generally comparable to each other; combining these two methods does not result in further gain in our study.

The rest of this paper is organized as follows. Section 2 introduces the datasets that neural models are transferred across; Section 3 details the neural architectures and experimental settings. We describe two approaches (INIT and MULT) to transfer learning in Section 4. We present experimental results in Sections 5–6 and have concluding remarks in Section 7.

2 Datasets

In our study, we conducted two series of experiments using six open datasets as follows.

- **Experiment I**: Sentence classification
  - `IMDB`. A large dataset for binary sentiment classification (`positive` vs. `negative`).[5]
  - `MR`. A small dataset for binary sentiment classification.[6]
  - `QC`. A (small) dataset for 6-way question classification (e.g., `location`, `time`, and `number`).[7]

---

[3]The acronyms refer to *semantic role labeling*, *named entity recognition*, *part-of-speech tagging*, and *chunking*, respectively.

[4]Here, we quote the accuracies obtained by using unsupervised pretraining of word embeddings. This is the highest performance in that paper; using pretrained word embeddings is also a common practice in the literature.

---

[5]https://drive.google.com/file/d/
0B8yp1gOBCztyN0JaMDVoeXhHWm8/
[6]https://www.cs.cornell.edu/people/pabo/
movie-review-data/
[7]http://cogcomp.cs.illinois.edu/Data/QA/QC/

Statistics (# of Samples)					
	Experiment I			Experiment II		
	IMDB	MR	QC	SNLI	SICK	MSRP
#Train	550,000	8,500	4,800	550,152	4,439	3,575
#Val	50,000	1,100	600	10,000	495	501
#Test	2,000	1,100	500	10,000	4,906	1,725

Examples in Experiment I	
Sentiment Analysis (IMDB and MR)	
An idealistic love story that brings out the latent 15-year-old romantic in everyone.	+
Its mysteries are transparently obvious, and its too slowly paced to be a thriller.	−

Question Classification (QC)	
What is the temperature at the center of the earth?	number
What state did the Battle of Bighorn take place in?	location

Examples in Experiment II		
Natural Language Inference (SNLI and SICK)		
Premise	Two men on bicycles competing in a race.	
Hypothesis	People are riding bikes.	E
	Men are riding bicycles on the streets.	C
	A few people are catching fish.	N

Paraphrase Detection (MSRP)	
The DVD-CCA then appealed to the state Supreme Court.	Paraphrase
The DVD CCA appealed that decision to the U.S. Supreme Court.	
Earnings per share from recurring operations will be 13 cents to 14 cents.	Non-Paraphrase
That beat the company's April earnings forecast of 8 to 9 cents a share.	

Table 1: Statistics and examples of the datasets.

- **Experiment II**: Sentence-pair classification

  - SNLI. A large dataset for sentence entailment recognition. The classification objectives are entailment, contradiction, and neutral.[8]
  - SICK. A small dataset with exactly the same classification objective as SNLI.[9]
  - MSRP. A (small) dataset for paraphrase detection. The objective is binary classification: judging whether two sentences have the same meaning.[10]

In each experiment, the large dataset serves as the source domain and small ones are the target domains. Table 1 presents statistics of the above datasets.

We distinguish two scenarios of transfer regarding semantic similarity: (1) semantically equivalent transfer (IMDB→MR, SNLI→SICK), that is, the tasks of $\mathcal{S}$ and $\mathcal{T}$ are defined by the same meaning,

---

[8]http://nlp.stanford.edu/projects/snli/

[9]http://http://alt.qcri.org/semeval2014/task1/

[10]http://research.microsoft.com/en-us/downloads/

and (2) semantically different transfer (IMDB→QC, SNLI→MSRP). Examples are also illustrated in Table 1 to demonstrate semantic relatedness.

It should be noticed that in image or speech processing (Yosinski et al., 2014; Wang and Zheng, 2015), the input of neural networks pretty much consists of raw signals; hence, low-level feature detectors are almost always transferable, even if Yosinski et al. (2014) manually distinguish artificial objects and natural ones in an image classification task.

Distinguishing semantic relatedness—which emerges from very low layers of either word embeddings or the successive hidden layer—is specific to NLP and also a new insight of our paper. As we shall see in Sections 5 and 6, the transferability of neural networks in NLP is more sensitive to semantics than in image processing.

3 Neural Models and Settings

In each group, we used a single neural model to solve three problems in a unified manner. That is to say, the neural architecture is the same among the three datasets, which makes it possible to investigate transfer learning regardless of whether the tasks are semantically equivalent. Concretely, the neural models are as follows.

- **Experiment I: LSTM-RNN**. To classify a sentence according to its sentiment or question type, we use a recurrent neural network (RNN, Figure 1a) with long short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). A softmax layer is added to the last word's hidden state for classification.

- **Experiment II: CNN-pair**. In this group, we use a "Siamese" architecture (Bromley et al., 1993) to classify the relation of two sentences. We first apply a convolutional neural network (CNN, Figure 1b) with a window size of 5 to model local context, and a max pooling layer gathers information to a fixed-size vector. Then the sentence vectors are concatenated and fed to a hidden layer before the softmax output.

In our experiments, embeddings were pretrained by word2vec (Mikolov et al., 2013); all embeddings and hidden layers were 100 dimensional. We
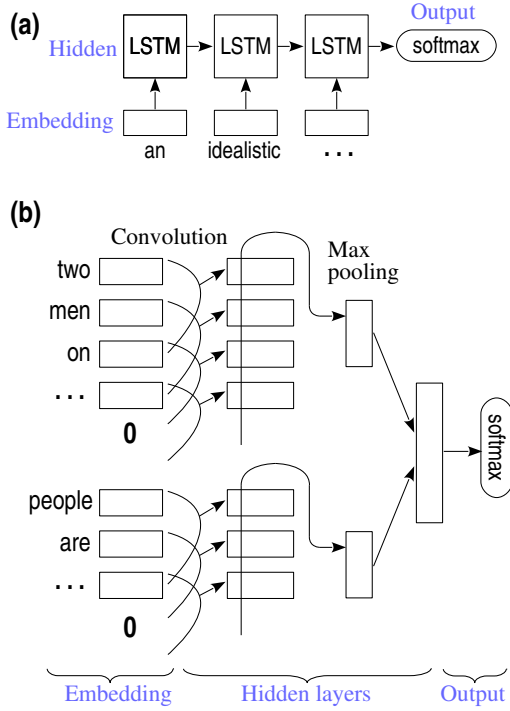
Figure 1: The models in our study. (a) Experiment I: RNNs with LSTM units for sentence classification. (b) Experiment II: CNN for sentence pair modeling.

applied stochastic gradient descent with a mini-batch size of 50 for optimization. In each setting, we tuned the hyperparameters as follows: learning rate from $\{3, 1, 0.3, 0.1, 0.03\}$, power decay of learning rate from {fast, moderate, low} (defined by how much, after one epoch, the learning rate residual is: $0.1x, 0.3x, 0.9x$, resp). We regularized our network by dropout with a rate from $\{0, 0.1, 0.2, 0.3\}$. Note that we might not run nonsensical settings, e.g., a larger dropout rate if the network has already been underfitting (i.e., accuracy has decreased when the dropout rate increases). We report the test performance associated with the highest validation accuracy.

To setup a baseline, we trained our models without transfer 5 times by different random parameter initializations (Table 2). We have achieved reasonable performance that is comparable to similar models reported in the literature with all six datasets. Therefore, our implementation is fair and suitable for further study of transfer learning.

It should be mentioned that the goal of this paper is not to outperform state-of-the-art results; instead,

	Dataset	Avg acc.±std.	Related model
Exp. I	IMDB	87.0	89.3 (Non-NN, Dong[+],2015)
	MR	75.1 ± 0.6	77.7 (RAE, Socher[+], 2013)
	QC	90.8 ± 0.9	90.2 (RNN, Zhao[+],2015)
Exp. II	SNLI	76.3	77.6 (RNN, Bowman[+],2015)
	SICK	70.9 ± 1.3	71.3 (RNN, Bowman[+],2015)
	MSRP	69.0 ± 0.5	69.6 (Arc-I CNN, Hu[+],2014)

Table 2: Accuracy (%) without transfer. We also include related models for comparison (Dong et al., 2015; Socher et al., 2011; Zhao et al., 2015; Bowman et al., 2015; Hu et al., 2014), showing that we have achieved comparable results, and thus are ready to investigate transfer learning. The models were run one only once in source domains, because we could only transfer a particular model instead of an average of several models.

we would like to conduct a fair comparison of different methods and settings for transfer learning in NLP.

4 Transfer Methods

Transfer learning aims to use knowledge in a source domain to aid the target domain. As neural networks are usually trained incrementally with gradient descent (or variants), it is straightforward to use gradient information in both source and target domains for optimization so as to accomplish knowledge transfer. Depending on how samples in source and target domains are scheduled, there are two main approaches to neural network-based transfer learning:

• Parameter initialization (INIT). The INIT approach first trains the network on $\mathcal{S}$, and then directly uses the tuned parameters to initialize the network for $\mathcal{T}$. After transfer, we may fix (🔒) the parameters in the target domain (Glorot et al., 2011), i.e., no training is performed on $\mathcal{T}$. But when labeled data are available in $\mathcal{T}$, it would be better to fine-tune (🔓) the parameters.

INIT is also related to unsupervised pretraining such as word embedding learning (Mikolov et al., 2013) and autoencoders (Bengio et al., 2006). In these approaches, parameters that are (pre)trained in an unsupervised way are transferred to initialize the model for a supervised task (Plank and Moschitti, 2013). However, our paper focuses on "supervised pretraining," which means we transfer knowledge from a labeled source domain.

482

- Multi-task learning (MULT). MULT, on the other hand, simultaneously trains samples in both domains (Collobert and Weston, 2008; Liu et al., 2016). The overall cost function is given by

$$J = \lambda J_{\mathcal{T}} + (1 - \lambda)J_{\mathcal{S}} \qquad (1)$$

where $J_{\mathcal{T}}$ and $J_{\mathcal{S}}$ are the individual cost function of each domain. (Both $J_{\mathcal{T}}$ and $J_{\mathcal{S}}$ are normalized by the number of training samples.) $\lambda \in (0, 1)$ is a hyperparameter balancing the two domains.

It is nontrivial to optimize Equation 1 in practice by gradient-based methods. One may take the partial derivative of J and thus λ goes to the learning rate (Liu et al., 2016), but the model is then vulnerable because it is likely to blow up with large learning rates (multiplied by λ or $1 - \lambda$) and be stuck in local optima with small ones.

Collobert and Weston (2008) alternatively choose a data sample from either domain with a certain probability (controlled by λ) and take the derivative for the particular data sample. In this way, domain transfer is independent of learning rates, but we may not be able to fully use the entire dataset of $\mathcal{S}$ if λ is large. We adopted the latter approach in our experiment for simplicity. (More in-depth analysis may be needed in future work.) Formally, our multi-task learning strategy is as follows.

1 Switch to $\mathcal{T}$ with prob. λ, or to $\mathcal{S}$ with prob. $1 - \lambda$.
2 Compute the gradient of the next data sample in the particular domain.

Further, INIT and MULT can be combined straightforwardly, and we obtain the third setting:

- Combination (MULT+INIT). We first pretrain on the source domain $\mathcal{S}$ for parameter initialization, and then train $\mathcal{S}$ and $\mathcal{T}$ simultaneously.

From a theoretical perspective, INIT and MULT work in different ways. In the MULT approach, the source domain regularizes the model by "aliasing" the error surface of the target domain; hence the neural network is less prone to overfitting. In INIT, $\mathcal{T}$'s error surface remains intact. Before training on the target dataset, the parameters are initialized in such a meaningful way that they contain additional knowledge in the source domain. However, in an extreme case where $\mathcal{T}$'s error surface is convex, INIT is ineffective because the parameters can reach the global optimum regardless of their initialization. In practice, deep neural networks usually have highly complicated, non-convex error surfaces. By properly initializing parameters with the knowledge of $\mathcal{S}$, we can reasonably expect that the parameters are in a better "catchment basin," and that the INIT approach can transfer knowledge from $\mathcal{S}$ to $\mathcal{T}$.

5 Results of Transferring by INIT

We first analyze how INIT behaves in NLP-based transfer learning. In addition to two different transfer scenarios regarding semantic relatedness as described in Section 2, we further evaluated two settings: (1) fine-tuning parameters 🔓, and (2) freezing parameters after transfer 🔒. Existing evidence shows that frozen parameters would generally hurt the performance (Peng et al., 2015), but this setting provides a more direct understanding on how transferable the features are (because the factor of target domain optimization is ruled out). Therefore, we included it in our experiments. Moreover, we transferred parameters layer by layer to answer our second research question.

Through Subsections 5.1–5.3, we initialized the parameters of $\mathcal{T}$ with the ones corresponding to the highest validation accuracy of $\mathcal{S}$. In Subsection 5.4, we further investigated when the parameters are ready to be transferred during the training on $\mathcal{S}$.

5.1 Overall Performance

Table 3 shows the main results of INIT. A quick observation is that, in both groups, transfer learning of semantically equivalent tasks (IMDB→MR, SNLI→SICK) appears to be successful with an improvement of ~6%. The results are not surprising and also reported in Bowman et al. (2015).

For IMDB→QC and SNLI→MSRP, however, there is no improvement of transferring hidden layers (embeddings excluded), namely LSTM-RNN units and CNN feature maps. The E🔓H🔓O□ setting yields a slight degradation of 0.2–0.4%, ~.5x std. The incapability of transferring is also proved by locking embeddings and hidden layers

(E🔒H🔒O□). We see in this setting, the test performance is very low in `QC` or even worse than majority-class guess in `MSRP`. By further examining its training accuracy, which is 48.2% and 65.5%, respectively, we conclude that extracted features by LSTM-RNN and CNN models in $\mathcal{S}$ are almost irrelevant to the ultimate tasks $\mathcal{T}$ (`QC` and `MSRP`).

Although in previous studies, researchers have mainly drawn positive conclusions about transfer learning, we find a negative result similar to ours upon careful examination of Collobert and Weston (2008), and unfortunately, their results may be somewhat misinterpreted. In that paper, the authors report transferring NER, POS, CHK, and pretrained word embeddings improves the SRL task by 1.91–3.90% accuracy (out of 16.54–18.40% error rate), but their gain is mainly due to word embeddings. In the settings that use pretrained word embeddings (which is common in NLP), NER, POS, and CHK together improve the SRL accuracy by only 0.04–0.21%.

The above results are rather frustrating, indicating for RQ1 that neural networks may not be transferable to NLP tasks of different semantics. Transfer learning for NLP is more prone to semantics than the image processing domain, where even high-level feature detectors are almost always transferable (Donahue et al., 2014; Yosinski et al., 2014).

5.2 Layer-by-Layer Analysis

To answer RQ2, we next analyze the transferability of each layer. First, we freeze both embeddings and hidden layers (E🔒H🔒). Even in semantically equivalent settings, if we further freeze the output layer (O🔒), the performance in both `IMDB→MR` and `SNLI→SICK` drops, but by randomly initializing the output layer's parameters (O□), we can obtain a similar or higher result compared with the baseline (E⊠H□O□). The finding suggests that the output layer is mainly specific to a dataset. Transferring the output layer's parameters yields little (if any) gain.

Regarding embeddings and hidden layers (in the settings E🔓H🔓O□/E🔓H□O□ vs. E⊠H□O□), the `IMDB→MR` experiment suggests both of embeddings and the hidden layer play an important role, each improving the accuracy by 3%. In `SNLI→SICK`, however, the main improvement lies in the hidden layer. A plausible explanation is that

Experiment I		
Setting	IMDB→MR	IMDB→QC
Majority	50.0	22.9
E⊠ H□ O□	75.1	90.8
E🔒 H□ O□	78.2	93.2
E🔒 H🔒 O□	78.8	55.6
E🔒 H🔒 O🔒	73.6	–
E🔓 H□ O□	78.3	92.6
E🔓 H🔓 O□	81.4	90.4
E🔓 H🔓 O🔓	80.9	–
Experiment II		
Setting	SNLI→SICK	SNLI→MSRP
Majority	56.9	66.5
E⊠ H□ O□	70.9	69.0
E🔒 H□ O□	69.3	68.1
E🔒 H🔒 O□	70.0	66.4
E🔒 H🔒 O🔒	43.1	–
E🔓 H□ O□	71.0	69.9
E🔓 H🔓 O□	76.3	68.8
E🔓 H🔓 O🔓	77.6	–

Table 3: Main results of neural transfer learning by INIT. We report test accuracies (%) in this table. E: embedding layer; H: hidden layers; O: output layer. ⊠: Word embeddings are pretrained by `word2vec`; □: Parameters are randomly initialized); 🔒: Parameters are transferred but frozen; 🔓: Parameters are transferred and fine-tuned. Notice that the E🔒H🔒O🔒 and E🔓H🔓O🔓 settings are inapplicable to `IMDB→QC` and `SNLI→MSRP`, because the output targets do not share same meanings and numbers of target classes.

in sentiment classification tasks (`IMDB` and `MR`), information emerges from raw input, i.e., sentiment lexicons and thus their embeddings, but natural language inference tasks (`SNLI` and `SICK`) address more on semantic compositionality and thus hidden layers are more important.

Moreover, for semantically different tasks (`IMDB→QC` and `SNLI→MSRP`), the embeddings are the only parameters that have been observed to be transferable, slightly benefiting the target task by 2.7x and 1.8x std, respectively.

5.3 How does learning rate affect transfer?

Bowman et al. (2015) suggest that after transferring, a large learning rate may damage the knowledge stored in the parameters; in their paper, they transfer the learning rate information (AdaDelta) from $\mathcal{S}$ to $\mathcal{T}$ in addition to the parameters.
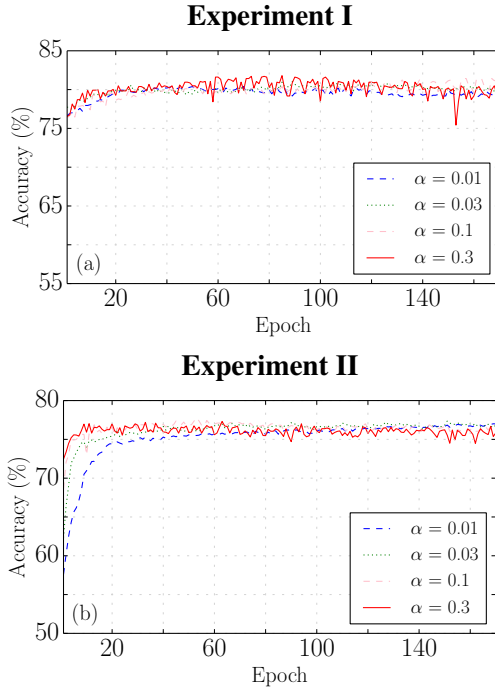
Figure 2: Learning curves of different learning rates (denoted as α). (a) Experiment I: IMDB→MR; (b) Experiment II: SNLI→SICK.

Although the rule of the thumb is to choose all hyperparameters—including the learning rate—by validation, we are curious whether the above conjecture holds. Estimating a rough range of sensible hyperparameters can ease the burden of model selection; it also provides evidence to better understand how transfer learning actually works.

We plot the learning curves of different learning rates α in Figure 2 (IMDB→MR and SNLI→SICK, E🔓H🔓O□). (In the figure, no learning rate decay is applied.) As we see, with a large learning rate like $\alpha = 0.3$, the accuracy increases fast and peaks at earlier epochs. Training with a small learning rate (e.g., $\alpha = 0.01$) is slow, but its peak performance is comparable to large learning rates when iterated by, say, 100 epochs. The learning curves in Figure 2 are similar to classic speed/variance trade-off, and we have the following additional discovery:

> In INIT, transferring learning rate information is not necessarily useful. A large learning rate does not damage the knowledge stored in the pretrained hyperparameters, but accelerates the training process to a large extent. In all, we may need to perform validation to choose the learning rate if computational resources are available.
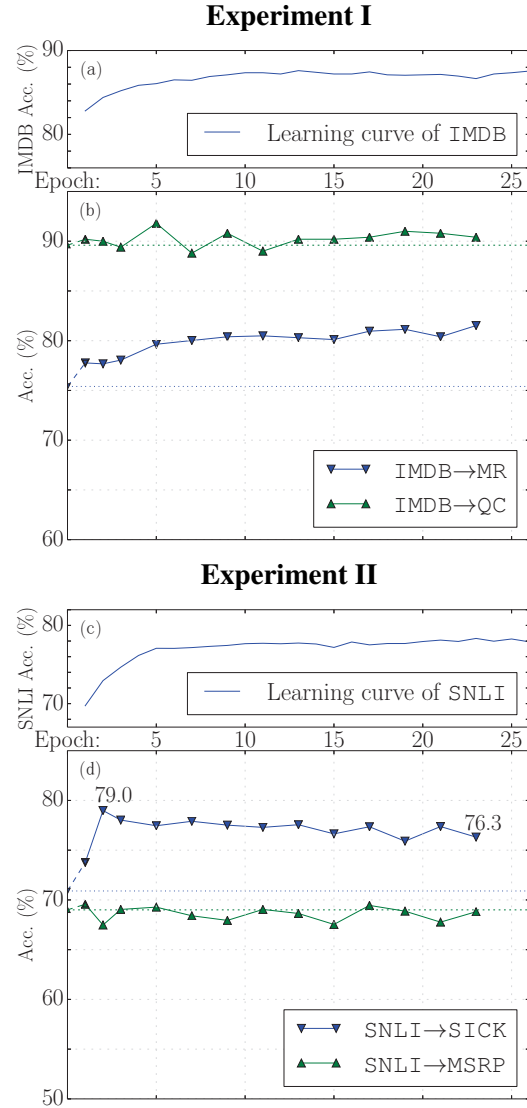

Figure 3: (a) and (c): Learning curves of $\mathcal{S}$. (b) and (d): Accuracies of $\mathcal{T}$ when parameters are transferred at a certain epoch during the training of $\mathcal{S}$. Dotted lines refer to non-transfer, which can be equivalently viewed as transferring before training on $\mathcal{S}$, i.e., epoch = 0. Note that the x-axis shares across different subplots.

5.4 When is it ready to transfer?

In the above experiments, we transfer the parameters when they achieve the highest validation performance on $\mathcal{S}$. This is a straightforward and intuitive practice.

However, we may imagine that the parameters well-tuned to the source dataset may be too specific to it, i.e., the model overfits $\mathcal{S}$ and thus may underfit $\mathcal{T}$. Another advantage of early transfer lies in com-

putational concerns. If we manage to transfer model parameters after one or a few epochs on $\mathcal{S}$, we can save much time especially when $\mathcal{S}$ is large.

We therefore made efforts in studying when the neural model is ready to be transferred. Figures 3a and 3c plot the learning curves of the source tasks. The accuracy increases sharply from epochs 1–5; later, it reaches a plateau but is still growing slowly.

We then transferred the parameters at different stages (epochs) of training to target tasks (also with the setting E◼🔒H◼🔒O□). Their accuracies are plotted in Figures 3b and 3d.

In IMDB→MR, the source performance and transferring performance align well. The SNLI→SICK experiment, however, produces interesting yet unexpected results. Using the second epoch of SNLI's training yields the highest transfer performance on SICK, i.e., 78.98%, when the SNLI performance itself is comparatively low (72.65% vs. 76.26% at epoch 23). Later, the transfer performance decreases gradually by ∼2.7%. The results in these two experiments are inconsistent and lack explanation.

6 MULT, and its Combination with INIT

To answer RQ3, we investigate how multi-task learning performs in transferring knowledge, as well as the effect of the combination of MULT and INIT. In this section, we applied the setting: sharing embeddings and hidden layers (denoted as E♥H♥O□), analogous to E◼🔒H◼🔒O□ in INIT. When combining MULT and INIT, we used the pretrained parameters of embeddings and hidden layers on $\mathcal{S}$ to initialize the multi-task training of $\mathcal{S}$ and $\mathcal{T}$, visually represented by E◼🔒♥H◼🔒♥O□.

In both MULT and MULT+INIT, we had a hyperparameter $\lambda \in (0, 1)$ balancing the source and target tasks (defined in Section 4). λ was tuned with a granularity of 0.1. As a friendly reminder, $\lambda = 1$ refers to using $\mathcal{T}$ only; $\lambda = 0$ refers to using $\mathcal{S}$ only. After finding that a small λ yields high performance of MULT in the IMDB+MR and SNLI+SICK experiments (thick blue lines in Figures 4a and 4c), we further tuned the λ from 0.01 to 0.09 with a fine-grained granularity of 0.02.

The results are shown in Figure 4. From the green curves in the 2nd and 4th subplots, we see MULT (with or without INIT) does not improve the accu-
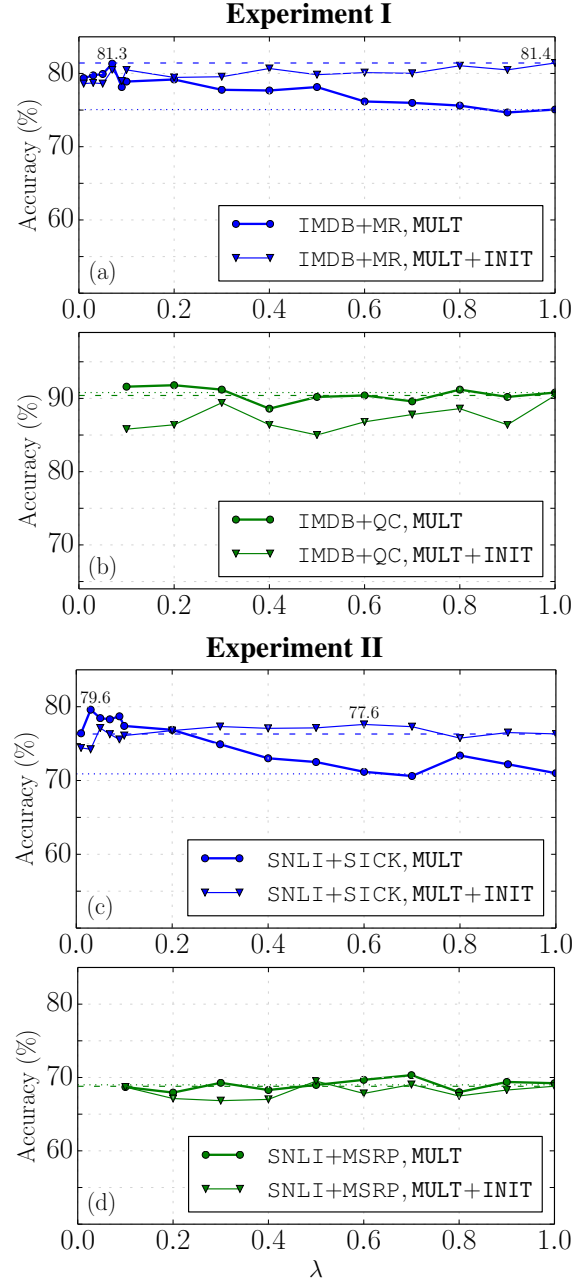

Figure 4: Results of MULT and MULT+INIT, where we share word embeddings and hidden layers. Dotted lines are the non-transfer setting; dashed lines are the INIT setting E◼🔒H◼🔒O□, transferred at the peak performance of IMDB and SNLI.

racy of target tasks (QC and MSRP); the inability to transfer is cross-checked by the INIT method in Section 5. For MR and SICK, on the other hand, transferability of the neural model is also consistently positive (blue curves in Figures 4a and 4c), supporting our conclusion to RQ1 that neural trans-

fer learning in NLP depends largely on how similar in semantics the source and target datasets are.

Moreover, we see that the peak performance of MULT is slightly lower than INIT in Experiment I (Figure 4a), but higher in Experiment II (Figure 4c); they are in the same ballpark.

In MULT+INIT (E◗♥H◗♥O□), the transfer performance of MULT+INIT remains high for different values of λ. Because the parameters given by INIT have already conveyed sufficient information about the source task, MULT+INIT consistently outperforms non-transferring by a large margin. Its peak performance, however, is not higher than MULT or INIT. In summary, we answer our RQ3 as follows: in our experiments, MULT and INIT are generally comparable; we do not obtain further gain by combining MULT and INIT.

7 Concluding Remarks

In this paper, we addressed the problem of transfer learning in neural network-based NLP applications. We conducted two series of experiments on six datasets, showing that the transferability of neural NLP models depends largely on the semantic relatedness of the source and target tasks, which is different from other domains like image processing. We analyzed the behavior of different neural layers. We also experimented with two transfer methods: parameter initialization (INIT) and multi-task learning (MULT). Besides, we reported two additional studies in Sections 5.3 and 5.4 (not repeated here). Our paper provides insight on the transferability of neural NLP models; the results also help to better understand neural features in general.

How transferable are the conclusions in this paper? We have to concede that empirical studies are subject to a variety of factors (e.g., models, tasks, datasets), and that conclusions may vary in different scenarios. In our paper, we have tested all results on two groups of experiments involving 6 datasets and 2 neural models (CNN and LSTM-RNN). Both models and tasks are widely studied in the literature, and not chosen deliberately. Results are mostly consistent (except Section 5.4). Along with analyzing our own experimental data, we have also collected related results in previous studies, serving as additional evidence in answering our research questions.

Therefore, we think the generality of this work is fair and that the conclusions can be generalized to similar scenarios.

Future work. Our work also points out some future directions of research. For example, we would like to analyze the effect of different MULT strategies. More efforts are also needed in developing an effective yet robust method for multi-task learning.

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 120–128.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing:

Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on Machine Learning*, pages 647–655.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*, 41(2):293–336.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 451–459.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*, pages 513–520.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2750–2756.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2106–2111.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1225–1237.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of 13th European Conference on Computer Vision*, pages 818–833.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *International Joint Conference on Artificial Intelligence*, pages 4069–4076.